# Fast similarity search and clustering of video sequences on the world-wide-web

Sen-ching S. Cheung, *Member, IEEE,* Avideh Zakhor, *Fellow, IEEE*

***Abstract*— We define similar video content as video sequences with almost identical content but possibly compressed at different qualities, reformatted to different sizes and frame-rates, undergone minor editing in either spatial or temporal domain, or summarized into keyframe sequences. Building a search engine to identify such similar content in the World-Wide Web requires (a) robust video similarity measurements, (b) fast similarity search techniques on large databases, and (c) intuitive organization of search results. In [1], we proposed a randomized technique called the video signature (ViSig) method for video similarity measurement. In this paper, we focus on the remaining two issues by proposing a feature extraction scheme for fast similarity search, and a clustering algorithm for identification of similar clusters. Similar to many other content-based methods, the ViSig method uses high-dimensional feature vectors to represent video. To warrant a fast response time for similarity searches on high dimensional vectors, we propose a novel non-linear feature extraction scheme on arbitrary metric spaces that combines the triangle inequality with the classical Principal Component Analysis (PCA). We show experimentally that the proposed technique outperforms PCA, Fastmap, Triangle-Inequality Pruning, and Haar wavelet on signature data. To further improve retrieval performance, and provide better organization of similarity search results, we introduce a new graph-theoretical clustering algorithm on large databases of signatures. This algorithm treats all signatures as an abstract threshold graph, where the distance threshold is determined based on local data statistics. Similar clusters are then identified as highly connected regions in the graph. By measuring the retrieval performance against a ground-truth set, we show that our proposed algorithm outperforms simple thresholding, single-link and complete-link hierarchical clustering techniques.**

***Index Terms*— dimension reduction, similarity search, clustering, web search, video signature**

## I. Introduction

With ever more popularity of video web-publishing, much video content is being mirrored, reformatted, modified and republished. Such redundancy creates problems for multimedia search engines in that the search results become cluttered with a large number of similar versions of the same content. This degrades the usability of the search engine as studies show that users seldom go beyond the first screen of search results [2]. Content providers are also interested in using search engines to identify similar versions of their video content for legal, contractual or other business related reasons. To detect similar video content on the web, we require an algorithm to be capable of measuring sequences of any length, be robust against temporal re-ordering, and extremely efficient to execute.

In [1], we have proposed a randomized algorithm called ViSig, aimed at measuring the fraction of visually similar frames shared between two sequences. ViSig produces a compact representation called video signature for every video sequence in the database. The size of a signature is determined by the desired accuracy rather than the length of the video. This makes ViSig suitable to handle sequences with vastly different durations such as those found on the web. Compared with video similarity measurement schemes based on warping distance [3]–[5], Hausdorff distance [6], and template matching [7], the ViSig method does not need to store and examine the entire length of video sequence for similarity measurement. There are also other video summarization techniques for similarity measurement such as those based on density parametrization [8]–[10] or keyframes produced by k-means or k-medoids [6]. Compared with these techniques, ViSig has much lower computational complexity as it requires only a single pass of a video sequence to compute its signature.

Using an efficient algorithm such as ViSig to measure video similarity is only the first step toward building a video search engine. Besides similarity measurement, one needs to support fast similarity search over potentially millions of video signatures, and provide an intuitive organization of the search results. In this paper, we first propose a similarity search algorithm that takes advantage of how a video signature is computed and the use of a metric function in comparing two video signatures. Second, we propose a hierarchical clustering algorithm for a large database of video signatures that is robust to sampling error introduced by ViSig. Before describing the details of our algorithms, we first review existing work on fast similarity search and clustering in the following two sections.

### A. Fast similarity search

When a user presents a query video signature to the search engine, the search engine needs to identify all similar signatures in the database of possibly millions of entries. The naive approach of sequential search is too slow to handle large databases and complex comparison functions. Elaborate data structures, collectively known as the Spatial Access Methods (SAM), have been proposed to facilitate similarity search [11]–[13]. Most of these methods, however, do not scale well to

This work was supported by the Air Force Office of Scientific Research (AFOSR) Grant F49620-00-1-0327 and by National Science Foundation (NSF) Grant ANI-9905799.

S.-C. Cheung was with the University of California, Berkeley, CA 94720 USA. He is now with the Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551 USA. (e-mail:sccheung@ieee.org)

A. Zakhor is with Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA. (e-mail: avz@eecs.berkeley.edu)

high dimensional metric spaces used in video search [14]. One strategy to mitigate this problem is to design a transformation to map the original metric space to a low-dimensional space where a SAM structure can be efficiently applied.

A good feature extraction mapping should be able to closely approximate distances in the high-dimensional space using the corresponding distances in the low-dimensional projected space. The most commonly used feature extraction mapping is Principal Component Analysis (PCA). PCA has been shown to be optimal in approximating Euclidean distance [15]; a myriad of techniques have also been developed for generating PCA on large datasets [16]. If the underlying metric is not Euclidean, PCA is no longer optimal and more general schemes are needed. Multidimensional Dimension Scaling (MDS) is the most general class of techniques for creating mappings that preserve a high-dimensional metric in a low-dimensional space [17]. MDS solves a non-linear optimization problem by searching for the mapping that best approximates all the high-dimensional pairwise distances between data points. In most occasions, MDS is too complex to be used for similarity search.

There exist other low computational complexity feature extraction mappings for metric spaces. One such technique is the Fastmap algorithm proposed by Faloutsos and Lin [18]. Fastmap is a heuristics algorithm that uses Euclidean distance to approximate a general metric. The time complexity for generating a Fastmap mapping is linear with respect to the size of the database. Another class of techniques constructs feature extraction mappings based on distances between high-dimensional vectors and a set of random vectors. These kinds of "random mappings" have been shown to possess certain favorable theoretical properties. Specifically, [19] and [20] have shown that a specific form of the random mappings can achieve the best possible approximation of high-dimensional distances. Unfortunately, such mappings are quite complex, and effectively compute all the pairwise distances. A more practical version has been proposed in [21] for approximating a metric used in protein sequence matching. An even simpler version, called Triangle-Inequality Pruning (TIP), has been proposed by Berman and Shapiro [22] for similarity search on image databases.

In this paper, we propose a novel feature extraction mapping that improves upon TIP by (a) taking into account both the upper and lower bounds of the triangle inequalities, and (b) achieving any target dimension using PCA. As we will demonstrate in Section III-B, our proposed scheme outperforms other techniques including pure PCA, TIP, Fastmap, and Haar wavelet [23].

### *B. Similarity clustering*

Our proposed similarity search technique enables us to efficiently compute distances between signatures in a large database. The next step is to use such information to identify clusters of similar video sequences. The goal is to present similarity search results in terms of similar clusters rather individual video sequences. This prevents the top search results from being cluttered with similar video sequences and thus provides more distinctive choices for users. Clustering experiments on web text documents show that the number of such similar clusters is likely to be very large and highly dependent on the data [24]. Many popular optimization-based clustering algorithms, such as k-means, require precise number of clusters as input [25, ch. 13]. To properly determine the number of clusters, a typical approach is to run k-means multiple times with different number of clusters as input, and then select the output clustering structure that achieves the highest score as measured by a cluster validation criterion [26, ch. 2.4]. As each clustering structure is independent of each other, it is computationally expensive to apply k-means to such a large dataset for so many times. Another popular class of clustering algorithms, called the hierarchical algorithms, can simultaneously produce a hierarchy of clusterings [25, ch. 13]. As it is not necessary to recompute the entire clustering structure when changing the number of clusters, hierarchical algorithms are more suitable for our application.

Given a large database of signatures, we treat each signature as a vertex of a graph, and form edges between two signatures if their distances, as returned by the fast similarity search, are less than a certain threshold. Hierarchical clustering algorithms then consider the graphs formed at different thresholds, and identify parts of the graphs as clusters based on their degree of connectivity. The simplest of such algorithms are the single-link and complete-link algorithms [25]. The single-link algorithm identifies all the connected components in the graph as clusters, and the complete-link uses complete subgraphs. Our experiments indicate that neither algorithm produces satisfactory clustering results. Single-link produces clusters that consist of totally irrelevant video sequences, while complete-link ignores many true similar clusters. There are other hierarchical algorithms, such as Ward or Weighted Pair Group Method Average (WPGMA), that represent compromises between single-link and complete-link [25]. These algorithms define, in an ad-hoc fashion, a distance function between two clusters. Nevertheless, none of these distance functions can adequately capture the properties of signatures and their distance measurement. As ViSig is a randomized algorithm, it is possible for two similar video sequences to produce signatures that are very far apart. In addition, the proposed fast similarity search may erroneously discard small distances due to the limitation of the feature extraction mapping. All in all, there is a need to design a more robust hierarchical clustering algorithm for signature data.

In this paper, we propose a new clustering algorithm that allows the user to adjust the level of connectivity for cluster identification. In the proposed algorithm, a connected component forms a cluster if its edge density exceeds a user-defined threshold. The algorithm is robust to errors in distance measurement, which are usually manifested as missing edges from a fully-connected connected component. Not only does this algorithm produces favorable retrieval results, but also it admits a simple implementation based on the classical Minimum Spanning Tree (MST) algorithm by Kruskal [27]. In [28], Zahn uses MST to separate data into different clusters if the MST branch connecting them is significantly longer than the nearby edges [28]. We extend this concept to also consider

the connectivity of the clusters.

The remainder of this paper is organized as follows: in Section II, we describe our proposed methods for (a) signature generation, (b) fast similarity search on signatures, and (c) signature clustering. Signature generation in Section II-A is a review of our early work in [1]. In Section II-A, we also introduce a procedure to transform similarity search in video signatures into a sequence of similarity search in a metric space. This is important because our proposed search algorithm in Section II-B depends on certain properties of a metric. The search algorithm in Section II-B also exploits how a video signature is generated in forming a low-dimensional "projection vector" used for fast search. Our proposed clustering algorithm in Section II-C does not depend on a metric and is applied directly on video signatures. Experimental results are provided in Section III to demonstrate the performance of our proposed techniques. Finally, we conclude this paper by discussing future work in Section IV.

## II. Algorithms

In Section II-A, we briefly review the ViSig method to generate signatures and describe a procedure to perform similarity search on a signature database. A novel algorithm for fast similarity search on signatures is described in Section II-B. Finally, we describe our proposed clustering algorithm in Section II-C.

### A. Overview of ViSig and Signature similarity search

We assume that individual frames in a video are represented by high-dimensional feature vectors from a metric space $(F, d(\cdot,\cdot))$[1]. To every frame, we associate a feature vector, and for convenience, we refer to the frame and its feature vector interchangeably. The metric $d(x, y)$ is used to measure the visual dissimilarity between feature vectors $x$ and $y$. $x$ and $y$ are visually similar if $d(x, y)$ is smaller than a small positive $\epsilon$, which may depend on the particular feature vectors. Each video sequence $X$ in the database is represented by a signature $X_S$, defined to be the collection of feature vectors in $X$ that are closest to a set of seed feature vectors $S = \{s_1, s_2, \ldots, s_m\}$:

$$X_S \triangleq (g_X(s_1), g_X(s_2), \ldots, g_X(s_m)) \quad (1)$$

where

$$g_X(s) \triangleq \arg\min_{x \in X} d(x, s)$$

The seed vectors are feature vectors randomly sampled from a clustered training set that is representative of the ensemble of target video data under consideration.

The central idea behind the ViSig method is that if two video clips share a large fraction of similar feature vectors, their signature vectors with respect to the same seed vectors are likely to be similar. Nonetheless, depending on the relative positions of the seed vectors and the frames, it is possible to sample non-similar signature vectors from two almost-identical video sequences. To mitigate such errors, in our previous work we have proposed a Ranking Function $Q(g_X(s))$ to determine the robustness of a signature vector $g_X(s)$ [1]:

$$Q(g_X(s)) \triangleq \min_{x \in X,\, d(x, g_X(s)) > \epsilon_C} d(x, s) - d(g_X(s), s). \quad (2)$$

where $\epsilon_C$ is the maximum distance between similar vectors within the video. We have shown that the larger $Q(g_X(s))$ is, the more robust $g_X(s)$ becomes. In order to guarantee the existence of robust signature vectors, we typically set the number of signature vectors in a signature, $m$, to be fairly large, but use only the most robust, or highest-ranked, $m'$ vectors when comparing two signatures. Specifically, two signature $X_S$ and $Y_S$ are compared using the following Signature Distance:

$$d_{sig}(X_S, Y_S) \triangleq \text{median}\{d(g_X(s_{j[i]}), g_Y(s_{j[i]})) : i = 1, 2, \ldots, m'\} \quad (3)$$

where $j[i]$ is the particular permutation of $i = 1, 2, \ldots, m$ such that $Q(g_X(s_{j[1]}) \geq Q(g_X(s_{j[2]})) \geq \ldots \geq Q(g_X(s_{j[m]}))$. In this paper, we declare two signatures to be similar if their signature distance is less than or equal to $\epsilon$. Different $\epsilon$ values are used for precision-recall trade-off in retrieval performance.

In Equation (3), we only use the ranking of $X_S$ and ignore that of $Y_S$. This is different from the definition found in [1], where the rankings of both signatures are used. Theoretically, such an asymmetry may lead to bias in the measurements. For example, if one video is a sub-sequence of the other, using the ranking of the shorter video may result in a smaller signature distance than using that of the longer one. Nonetheless, we have found little difference in the experimental results between the two definitions. We opt for this asymmetric definition because it reduces the signature similarity search problem into a metric-space similarity search problem, which is fundamental to many fast algorithms studied in the literature.

In the signature similarity search problem, we want to identify all the signatures $Y_S$ in the database with $d_{sig}(X_S, Y_S) \leq \epsilon$, where $X_S$ is the input query. However, $d_{sig}(\cdot)$, as defined in Equation (3), is not a metric because $d_{sig}(X_S, Y_S)$ can be zero even if $X_S$ is not identical to $Y_S$. Nevertheless, the signature similarity search problem can be reduced to metric-space similarity search problems using the following procedure:

*Procedure 2.1 (Signature Similarity Search):*

1) Let $X_S = (g_X(s_1), \ldots, g_X(s_m))$ be the query signature, and $j[1], \ldots, j[m]$ be the corresponding ranking.
2) For each of the top $m'$ ranked signature vectors $g_X(s_{j[i]})$, identify those signature vectors $g_Y(s_{j[i]})$ in the database with $d(g_X(s_{j[i]}), g_Y(s_{j[i]})) \leq \epsilon$. This is a similarity search problem in the metric space $(F, d(\cdot))$. Let $E$ be the collection of all the signatures with at least one of their signature vectors returned in these metric-space similarity searches, i.e.

$$E \triangleq \{Y_S : d(g_X(s_{j[i]}), g_Y(s_{j[i]})) \leq \epsilon \text{ for some } i = 1, 2, \ldots, m'\}$$

3) Return all the signatures $Y_S$ in $E$ which have more than half of their signature vectors within $\epsilon$ of the signature

[1]For all $x, y$ in a metric space $F$, the function $d(x, y)$ is a metric if a) $d(x, y) \geq 0$; b) $d(x, y) = 0 \Leftrightarrow x = y$; c) $d(x, y) = d(y, x)$; d) $d(x, y) \leq d(x, z) + d(z, y)$, for all $z$.

vectors in $X_S$. This is equivalent to finding all the $Y_S$ in the database with $d_{sig}(X_S, Y_S) \leq \epsilon$.

In step two, by using only the ranking of $X_S$, rather than that of the $Y_S$'s in the database, we turn the signature similarity search problem into $m'$ similarity search problems in the metric space $(F, d(\cdot))$. The output set $E$ is the set of signatures in the database that share at least one similar signature vector with the query. The size of $E$ is typically small for a highly heterogeneous database such as the web. Thus, the last step of Procedure 2.1 is a simple search on a small set of signatures. As a result, in the remainder of this section, we shall focus on step two, i.e. the metric-space similarity search.

### *B. Fast similarity search on Signatures*

Our similarity search algorithm is based on a general framework called the GEMINI [12], reviewed in Section II-B.1. We also describe how we measure the performance of specific fast search algorithms in the same section. The key step in GEMINI, called the feature extraction step, is to design a mapping from the metric space of feature vectors to a very low-dimensional space where similarity search can be carried out efficiently. We propose a novel feature extraction mapping to be used in GEMINI for signature data in Section II-B.2.

*1) The GEMINI approach for similarity search:* GEneric Multimedia INdexIng, or GEMINI, is an approach for fast similarity search on data endowed with a metric function. Our description of GEMINI here is based on [12, ch.7]. Given a query vector $x$ and a database $D$ of feature vectors, we define the resulting set $A(x;\epsilon)$ of the similarity search on $x$ as follows:

$$A(x;\epsilon) \triangleq \{y \in D : d(x,y) \leq \epsilon\}. \tag{4}$$

Obviously, we can compute $A(x;\epsilon)$ by a sequential search on the database to identify those which are within $\epsilon$ of $x$. The GEMINI approach, as outlined below, attempts to avoid the complexity of a sequential search by projecting the vectors into a low-dimensional metric space where similarity search can be efficiently performed:

*Procedure 2.2 (GEMINI):*

1) Design a feature extraction mapping $\mathcal{T}$ which maps feature vectors from the metric space $(F, d(\cdot))$ to a very low dimensional metric space $(F', d'(\cdot))$. We call the vectors in $F'$ the Range Vectors and $d'(\cdot)$ the Range Metric.
2) For every feature vector $y$ in a database $D$, compute the corresponding range vector $\mathcal{T}(y)$ and store it in a SAM structure.
3) Given an input query feature vector $x$, first compute $\mathcal{T}(x)$, and then utilize the SAM structure computed in step 2 to perform a similarity search on $\mathcal{T}(x)$. The distance threshold used in this similarity search, which we refer to as Pruning Threshold and denote by $\epsilon'$, depends on both $\epsilon$ and $\mathcal{T}$. We will show how $\epsilon'$ is determined experimentally in Section III-B. The set of feature vectors that correspond to the result of this similarity search is called the Candidate Set:

$$C(x;\epsilon') \triangleq \{y \in D : d'(\mathcal{T}(x), \mathcal{T}(y)) \leq \epsilon'\} \tag{5}$$

4) It is possible that some feature vectors in $C(x;\epsilon')$ are far away from $x$. To complete the search, we sequentially compute the full metric function $d(\cdot)$ between $x$ and each of the vectors in $C(x;\epsilon')$, and identify those that are truly within $\epsilon$ of $x$. We denote this resulting set as $A'(x;\epsilon,\epsilon')$:

$$A'(x;\epsilon,\epsilon') \triangleq \{y \in C(x;\epsilon') : d(x,y) \leq \epsilon\}. \tag{6}$$

We can easily extend the GEMINI approach to handle similarity search on signatures by using Procedure 2.1 discussed in Section II-A. In the signature case, we denote the candidate set, the resulting set from sequential search, and the GEMINI resulting set as $C_S(X_S;\epsilon')$, $A_S(X_S;\epsilon)$, and $A'_S(X_S;\epsilon,\epsilon')$ respectively.

GEMINI solves the signature similarity search problem exactly if $A'_S(X_S;\epsilon,\epsilon')$ is identical to $A_S(X_S;\epsilon)$. GEMINI is more efficient than sequential search if the following two conditions hold:

1) The dimension of the range space is small. The dimension directly affects the speed of similarity search on range vectors in the following two aspects: first, a low-dimensional metric is typically faster to compute than the high-dimensional one; second, as shown in [14], we can expect a typical SAM structure to deliver faster-than-sequential search time if the dimension is below ten.
2) A typical candidate set is small enough so that few full metric computations are required in the last step of GEMINI.

In this paper, we assume the first condition holds, and do not delve into details of the design and implementation of any particular SAM structure, as it has been extensively studied elsewhere [11]–[13]. Instead, we focus on the design of a feature extraction mapping $\mathcal{T}$ to achieve the second condition: making the candidate set as small as possible. Based on the definition in Equation (5), a candidate set can be made arbitrarily small by decreasing the pruning threshold $\epsilon'$. Nonetheless, $\epsilon'$ cannot be too small, or else most or all of the correct retrievals in $A_S(X_S;\epsilon)$ may be excluded. As a result, there is a trade-off between the complexity, as measured by the size of the candidate set, and the accuracy, as measured by the fraction of the correct retrievals retained. Specifically, we define two quantities, Accuracy and Pruning, to quantify this trade-off, and use them to evaluate the performances of different feature extraction mappings.

Let $R$ be a typical set of query signatures. Accuracy is defined as the ratio between the sizes of the resulting sets obtained by GEMINI and by sequential search:

$$\text{Accuracy}(\epsilon,\epsilon') \triangleq \frac{\sum_{X_S \in R} |A'_S(X_S;\epsilon,\epsilon')|}{\sum_{X_S \in R} |A_S(X_S;\epsilon)|}, \tag{7}$$

where $|A|$ denotes the cardinality of set $A$. Since $A'_S(X_S;\epsilon,\epsilon')$ contains only those signatures in $C_S(X_S;\epsilon')$ that are similar to $X_S$, its size is always smaller than or equal to that of $A_S(X_S;\epsilon)$ which contains all the signatures in the database similar to $X_S$. As a result, the dynamic range of Accuracy is from zero to one, with one corresponding to the perfect

retrieval. The complexity is measured by Pruning, which is defined as the relative difference in the numbers of full signature distance operations between GEMINI and sequential search:

$$\text{Pruning}(\epsilon') \triangleq \frac{|R| \cdot |D| - \sum_{X_S \in R} |C_S(X_S; \epsilon')|}{|R| \cdot |D|} \quad (8)$$

The dynamic range of pruning is also between zero and one, with one corresponding to the maximum reduction in complexity. We can explain the numerator and denominator in Equation (8) as follows: the number of signature distance operations required by the sequential search is the product of the number of queries and the size of the database, i.e. $|R| \cdot |D|$. For GEMINI, full signature distance operations are only required in step 4 of Procedure 2.2, and their total number amounts to the combined size of all candidate sets, i.e. $\sum_{X_S \in R} |C_S(X_S; \epsilon')|$. As we assumed earlier that similarity search on range vectors can be efficiently handled by the SAM structure, we ignore the contribution from step 3 of Procedure 2.2 in the definition of Pruning.

*2) Feature Extraction for Signature:* The following procedure describes our proposed feature extraction mapping $\mathcal{T}(x_s)$:

*Procedure 2.3 (Feature Extraction Mapping):*

1) Let $x_s$ be any vector from a database of signature vectors with respect to the seed vector $s$. Compute the Projection Vector $\mathcal{P}(x_s)$ as follows:

$$\mathcal{P}(x_s) \triangleq (d(x_s, s_1)^2, d(x_s, s_2)^2, \ldots, d(x_s, s_m)^2) \quad (9)$$

   Note that the distance between $x_s$ and each of the seed vectors $s_i$ used in Equation (9) has been computed earlier during the signature generation process as described in Equation (1).
2) Reduce the dimension of $\mathcal{P}(x_s)$ to any desirable value by applying PCA onto $D$. The resulting low-dimensional vector is the target feature extraction mapping $\mathcal{T}(x_s)$.

Procedure 2.3 consists of two steps. The first step forms a projection vector based on distance measurement between a signature vector and all seed vectors, followed by a second step of PCA which maps the projection vector into any target dimension. The reason for using a two-step approach is to make our algorithm general enough for arbitrary metric space data. By arbitrary we mean those that are not necessarily finite dimensional, or use Euclidean distance. Many real-life data such as protein sequences or text documents cannot be adequately represented in a finite dimensional vector space, even though their similarity can often be measured via a properly defined metric function. PCA by itself can only be applied to a finite-dimensional vector space, and is optimal for Euclidean distance. The first step of our approach defines the projection-vector mapping based only on metric distance. This is in contrast with PCA which requires a coordinate system and a definition of inner product. As a result, our approach can be applied to arbitrary metric spaces, making it more general than PCA. The main innovation of Procedure 2.3 is the use of the projection vector mapping described in Equation (9). In the remainder of this section, we discuss the motivation behind the use of projection vectors.

Let $S$ be the set of seed vectors given by $\{s_1, s_2, \ldots, s_m\}$. Let $x_s$ and $y_s$ be the signature vectors in signatures $X_S$ and $Y_S$ that correspond to the same seed vector $s \in S$. Consider the following $m$-dimensional vector,

$$\mathcal{T}^*(x_s) \triangleq (d(x_s, s_1), d(x_s, s_2), \ldots, d(x_s, s_m)), \quad (10)$$

as a feature extraction mapping of $x_s$. We are interested in this particular formulation because of the following two reasons: first, this mapping does not require complex metric computation as it is based on $d(x_s, s_i)$ for $i = 1, \ldots, m$, quantities that are readily available from signature generation as described in Equation (9). Second, the distance $d(x_s, y_s)$ between any two signature vectors $x_s$ and $y_s$ can be related to the coordinates of the corresponding range vectors $\mathcal{T}^*(x_s)$ and $\mathcal{T}^*(y_s)$ by the triangle inequality:

$$|d(x_s, s_i) - d(y_s, s_i)| \leq d(x_s, y_s) \leq d(x_s, s_i) + d(y_s, s_i) \quad (11)$$

for $i = 1, 2, \ldots, m$. The above inequalities are instrumental in designing the feature extraction mapping. Our goal is to make use of these inequalities to design a range metric function $d'(\cdot)$ such that small $d(x_s, y_s)$ values correspond to "small" $d'(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s))$ values and vice versa.

The mapping $\mathcal{T}^*(\cdot)$ and its variations have been previously proposed in the literature for feature extraction [19]–[22]. These techniques typically use a $l_p$-metric [25, p. 361] as the range metric between $\mathcal{T}^*(x_s)$ and $\mathcal{T}^*(y_s)$. $l_p$-metric for $p = 1, 2, \ldots$ is given by

$$l_p(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \triangleq \left[ \sum_{i=1}^{m} \frac{|d(x_s, s_i) - d(y_s, s_i)|^p}{m} \right]^{1/p} \quad (12)$$

The special case of $l_\infty(\cdot)$ is defined as follows:

$$l_\infty(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \triangleq \max_{i=1,2,\ldots,m} |d(x_s, s_i) - d(y_s, s_i)| \quad (13)$$

We use a normalization factor of $1/m$ in the definition of $l_p$ in Equation (12) so that it is of the same order of magnitude as the $l_\infty$-metric. All the different variations of $l_p$ metric functions are composed of different powers of the absolute differences between the coordinates of $\mathcal{T}^*(x_s)$ and $\mathcal{T}^*(y_s)$, i.e. $|d(x_s, s_i) - d(y_s, s_i)|$, for $i = 1, 2, \ldots, m$. These absolute differences, however, appear only in the lower-bound half of the triangle inequality in (11). The consequence is that both metric functions defined in Equations (12) and (13) are upper-bounded by $d(x_s, y_s)$:

$$\begin{aligned} l_p(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) &\leq l_\infty(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \\ &\leq d(x_s, y_s) \end{aligned} \quad (14)$$

For any $x_s$ and $y_s$ with $d(x_s, y_s) \leq \epsilon$, Equation (14) implies that $l_\infty(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \leq \epsilon$ and $l_p(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \leq \epsilon$. This is an important property called the contractive property of the lower bound [12]. Consider the situation when $\mathcal{T}^*(\cdot)$ is used as the feature extraction mapping and $l_\infty$ or $l_p$ as the range metric in Procedure 2.2. By setting $\epsilon' = \epsilon$, the contractive property guarantees that the candidate set $C(x_s; \epsilon')$ contains the entire result set $A(x_s; \epsilon)$, thus achieving perfect accuracy. We call the approach of using $\mathcal{T}^*(\cdot)$ with $l_\infty$ as the

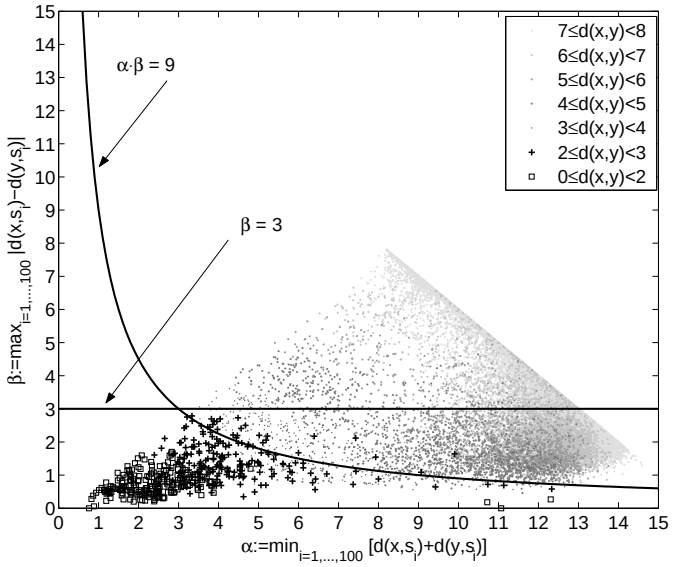


Fig. 1. Distribution of the metric $d(x, y)$ for 100,000 random pairs of signature vectors. The y-axis is $\max_{i=1,2,\ldots,100} |d(x, s_i) - d(y, s_i)|$ and the x-axis is $\min_{i=1,2,\ldots,100}[d(x, s_i) + d(y, s_i)]$. Metric values larger than three are shown as dots of different shades. Metric values smaller than or equal to three are shown as squares and crosses.

feature extraction mapping in Procedure 2.2 the "lower-bound" method. $l_\infty$ is preferred over other $l_p$ metrics as it is a closer estimate to $d(x_s, y_s)$ as indicated in Equation (14).

Nevertheless, for similarity search on the web, fast search time is typically more desirable than perfect accuracy. Using a simple experiment, we now demonstrate that better pruning-accuracy trade-off can be achieved by combining both the upper and lower bounds of the triangle inequality. Our experiment is based on a random set of 10,000 signature vectors selected from a large web video dataset, and their distances with respect to a set of $m = 100$ seed vectors. The feature vectors are color histograms of selected frames from the video sequences and the distance function is $l_1$. More details about the use of color histogram and the dataset will be provided in Section III-A. For each pair of signature vectors $x_s$ and $y_s$, we compute $d(x_s, y_s)$, and illustrate its distribution in Figure 1 as a function of a single lower and upper bound, defined as follows: for the lower bound, we take the maximum over all the individual lower bounds in (11), which is identical to $l_\infty(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s))$. For the upper bound, we use a similar approach and take the minimum of the individual upper bounds in (11) to form an $\alpha(\cdot)$ function:

$$\alpha(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \triangleq \min_{i=1,2,\ldots,m} [d(x_s, s_i) + d(y_s, s_i)] \quad (15)$$

Different colored points in Figure 1 correspond to different ranges of values for $d(x_s, y_s)$.

If we set $\epsilon$ to be 3.0, which we experimentally find to be a reasonable value to identify the majority of visually similar web video in the dataset, then metric values found in a typical resulting set $A(x_s; \epsilon)$ will include those black and magenta data points in Figure 1. Our goal then is to separate these "small-metric" points, from the rest of the "large-metric" points. If we use $l_\infty(\cdot)$ as the range metric, a typical candidate set based on the inequality $l_\infty(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \leq \epsilon'$ will include all the points below a horizontal line at level $\epsilon'$. An example of such a set with $\epsilon' = 3$ is shown in Figure 1. Even though all the small-metric points are within the candidate set, many of the large-metric points are also erroneously included as they have small $l_\infty(\cdot)$ values. It is clear, based on the shape of the distribution of the small-metric points, that a better separating function should combine both $l_\infty(\cdot)$ and $\alpha(\cdot)$. One possible choice is based on their product, $\beta(\cdot)$, defined as:

$$\beta(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \triangleq \quad (16)$$

$$\alpha(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \cdot l_\infty(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \quad (17)$$

As shown in the figure, even though the candidate set defined by $\beta(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \leq 9$ misses a few small-metric points, it excludes a much larger set of large-metric points than $l_\infty(\cdot)$. The problem with $\beta(\cdot)$ is that it is not a metric[2]. As a result, we cannot directly employ $\beta(\cdot)$ as our range metric.

The $\beta(\cdot)$ function in (17) is defined as the product of $\alpha(\cdot)$ and $l_\infty(\cdot)$, which represent the aggregate bounds of all the inequalities in (11). Rather than using the two aggregate bounds, it is much easier to form a metric by using the product of the bounds from the individual inequalities as follows:

$$[d(x_s, s_i) + d(y_s, s_i)] \cdot |d(x_s, s_i) - d(y_s, s_i)| = |d(x_s, s_i)^2 - d(y_s, s_i)^2| \quad (18)$$

for $i = 1, 2, \ldots, m$. Note that Equation (18) is in the form of an absolute difference. While absolute differences also appear in the definitions of $l_p$ metrics in Equation (12), the one in Equation (18) is the absolute difference of *squares* of $\mathcal{T}^*(\cdot)$'s coordinates, rather than absolute difference of coordinates of $\mathcal{T}^*(\cdot)$ themselves. Thus, it is conceivable to propose a new metric $\zeta(\cdot)$ that combines $l_2$ with this absolute difference of *squares* of coordinates as follows[3]:

$$\zeta(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s)) \triangleq \left( \frac{1}{m} \cdot \sum_{i=1}^{m} [d(x_s, s_i)^2 - d(y_s, s_i)^2]^2 \right)^{1/2} \quad (19)$$

Note that this metric applied to $\mathcal{T}^*(\cdot)$ can also be interpreted as the $l_2(\mathcal{P}(x_s), \mathcal{P}(y_s))$, where $\mathcal{P}(x_s)$ is the projection vector defined in Equation (9). Therefore, $l_2$ metric with the projection vector mapping is equivalent to applying the $\zeta(\cdot)$ metric onto $\mathcal{T}^*(\cdot)$ mapping.

In Section III-B, we will experimentally demonstrate that the projection vector mapping with $l_2$-metric outperforms both the "lower-bound" scheme using $l_\infty(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s))$ and the "product" scheme using $\beta(\mathcal{T}^*(x_s), \mathcal{T}^*(y_s))$. In practice, the $m$-dimensional projection vector may still be too large for most SAM structures. In step 2 of our proposed Procedure 2.3, we apply PCA to projection vectors in order to produce the best approximation of $l_2(\mathcal{P}(x_s), \mathcal{P}(y_s))$ for any given target

[2] $\beta(\cdot)$ is not a metric because, for an arbitrary pair of $m$-dimensional vectors $u$ and $v$, $\beta(u, v)$ becomes zero when $u$ and $v$ share a zero in any one of the coordinates, rather than $u = v$ as required by a true metric.

[3] Technically, $\zeta(\cdot)$ forms a metric only for real vectors with non-negative (or non-positive) coordinates, which is the case for our $\mathcal{T}^*(\cdot)$ vectors. If both positive and negative coordinates are allowed, $\zeta(\cdot)$ fails to become a metric as $\zeta(u, v) = 0$ when the coordinates $u$ and $v$ have the same magnitudes but different signs.

dimension. Experimental results in Section III-B show the superior performance of Procedure 2.3 over other dimension reduction schemes.

### C. Similarity clustering on Signatures

By applying the fast similarity search techniques described in Section II-B, it is possible to rapidly identify video sequences similar to a given query in a large database. In this section, we introduce a clustering algorithm that utilizes such information to identify all clusters of similar video sequences in the database.

In order to describe our clustering algorithm in a general framework, we will treat the set of signatures and the distance relationship as a graph. A graph $\mathcal{G}$ has a set of vertices $V(\mathcal{G})$ and a set of edges $E(\mathcal{G}) \subset V(\mathcal{G}) \times V(\mathcal{G})$. All edges we consider are undirected. In many occasions, we only consider a portion of a graph. Thus, we need the notion of a subgraph – $\mathcal{G}'$ is a subgraph of $\mathcal{G}$ if $V(\mathcal{G}') \subset V(\mathcal{G})$ and $E(\mathcal{G}') \subset E(\mathcal{G})$. If the subgraph $\mathcal{G}'$ inherits all the edges between its vertices from the original graph, $\mathcal{G}'$ is called an induced subgraph. In our application, each signature in the database is represented as a vertex in a graph. Ideally, two signatures should be linked with an edge if the two corresponding video clips are truly similar. Typically, a small signature distance is a good indicator of an edge. Nevertheless, even if the signature distance between two vertices is large, there might still be a need for an edge if, for example, there has been an error in the distance measurement. Such an error may be revealed if there are many other signatures that are simultaneously close to both of them. The primary goal of a clustering algorithm is to make use of all measured distance relationship to infer the most reasonable placement of edges. Since it is computationally infeasible to search all possible graphs, we only consider a special subset of them called the threshold graphs. A threshold graph $P(\mathcal{V}, \rho)$ has a vertex set $\mathcal{V}$, and has an edge between every two of its vertices if the distance between them is less than $\rho > 0$. The length of an edge is defined to be the signature distance in Equation (3). We ignore the asymmetry in Equation (3) and treat every edge as undirected.

In the absence of any error in distance measurement and similarity search, we assume that the largest possible signature distance between two truly similar video clips is $\mu$. The choice of $\mu$ depends on the feature vector as well as the data. For the feature vector to be useful in similarity detection, $\mu$ is typically much smaller than the maximum distance value, and the threshold graph $P(\mathcal{V}, \mu)$ is very sparse. Our algorithm considers all the threshold graphs $P(\mathcal{V}, \rho)$ with $\rho \leq \mu$. Since each of these $P(\mathcal{V}, \rho)$ is very sparse, they are likely to contain many isolated Connected Components. A connected component, or CC, of a graph is an induced subgraph in which all vertices are reachable from each other, but completely disconnected from the rest of the graph. CC's in the threshold graph $P(\mathcal{V}, \rho)$ are prime candidates for similar clusters: all signatures in a CC $\mathcal{C}$ are at least $\rho$ away from the rest of the database. If $\mathcal{C}$ is also a complete graph, which means that there is an edge between every pair of signatures, intuitively it corresponds to what a similar cluster should be : all video sequences in it are similar to each other but far away from the rest of the database. In practice, full completeness is too stringent of a requirement to impose because the randomness in signatures may erroneously amplify the distance between some similar video sequences. Thus, as long as $\mathcal{C}$ is close to a complete subgraph, it is reasonable to assume that it represents a similar cluster. To this end, we need a measurement of the density of edges inside a CC. Note that for any CC $\mathcal{C}$, there are at least $|V(\mathcal{C})| - 1$ edges as $\mathcal{C}$ is connected, and at most $|V(\mathcal{C})| \cdot (|V(\mathcal{C})| - 1)/2$ edges if there is one edge between every pair of vertices. We can thus define an edge density function $\Gamma(\mathcal{C})$ between these two extremes as:

$$\Gamma(\mathcal{C}) \triangleq \begin{cases} \frac{|E(\mathcal{C})| - (|V(\mathcal{C})| - 1)}{|V(\mathcal{C})| \cdot \frac{(|V(\mathcal{C})|-1)}{2} - (|V(\mathcal{C})|-1)} & \text{if } |V(\mathcal{C})| > 2 \\ 1 & \text{otherwise,} \end{cases} \tag{20}$$

$\Gamma(\mathcal{C})$ evaluates to 0 when $\mathcal{C}$ is barely connected, and to 1 when $\mathcal{C}$ is complete. We define a similar cluster to be a CC whose edge density exceeds a fixed threshold $\gamma \in (0, 1]$.

We are now ready to describe our clustering algorithm: given a database of signatures $\mathcal{V}$, we compute $P(\mathcal{V}, \mu)$ by performing a fast similarity search on each signature to identify all those that are within distance $\mu$ away. The resulting $P(\mathcal{V}, \mu)$ is composed of CC's with varying edge densities. Those CC's with edge densities larger than $\gamma$ are identified as similar clusters and removed from the graph. For the remaining CC's, we start removing edges in decreasing order of length until some similar clusters emerge. To avoid bias, edges of the same length are removed simultaneously. This process of edge removal is equivalent to lowering the distance threshold $\rho$ until the graph is partitioned into multiple CC's. CC's with high enough edge densities are identified as similar clusters. This process of lowering distance threshold and checking edge density is repeated until we exhaust all the CC's.

The key step of the above algorithm is to find the appropriate distance threshold to partition a CC $\mathcal{C}$ once it is found not be a similar cluster. A naive approach is to check whether $\mathcal{C}$ remains connected after recursively removing the longest edge, or edges if there are multiple of them with the same length, from $\mathcal{C}$. This approach is computationally expensive as we need to check connectedness after each edge removal. Let $\mathcal{E}$ be the set of equally-long edges last removed from $\mathcal{C}$ before it is separated into a number of new CC's. It can be shown that the longest branch in any Minimum Spanning Tree (MST) of $\mathcal{C}$ must belong to $\mathcal{E}$ [29, Appx. 5B]. A MST $\mathcal{T}$ of a connected graph $\mathcal{C}$ is a subgraph that connects all vertices with the least sum of edge length. Thus, the length of the longest branch in $\mathcal{T}$ represents the appropriate distance threshold to partition $\mathcal{C}$ into CC's. Furthermore, after partitioning $\mathcal{C}$ into a new set of CC's, the subtree of $\mathcal{T}$ in each newly-formed CC is also a MST. As a result, we can repeat the same process for each CC without ever recomputing a new MST. In a nutshell, we only need to compute the MST once for the original threshold graph $P(\mathcal{V}, \mu)$, and the distance threshold required for partitioning any CC in the graph is the length of one of the branches in the MST. We now summarize the procedure of our clustering

algorithm as follows[4]:

*Procedure 2.4 (Signature Similarity Clustering):*

1) Compute $P(\mathcal{V}, \mu)$ by applying fast similarity search on each node in $\mathcal{V}$.
2) MST Construction
   a) Sort all edges in $P(\mathcal{V}, \mu)$ in increasing order of length.
   b) For each edge $e$ in the sorted list, determine if $e$ is a MST branch.
   c) If $e$ is not a MST branch, it must belong to one of the CC's. Increment the edge and node counts of the CC that contains $e$.
   d) If $e$ is a MST branch, compute the edge densities of the two CC's joined by $e$ based on their respective edge and node counts.
3) Cluster Formation
   a) Examine each branch $t$ of the MST in decreasing order of length. No sorting is required because edges are already sorted in step 2(a).
   b) If the CC attached to $t$ has an edge density larger than or equal to $\gamma$, declare it as a similar cluster and remove it from the graph.

The second step of Procedure 2.4, i.e. MST Construction, basically follows the Kruskal algorithm described in [30, ch. 23]. The only difference is that whenever a new branch is added to the MST, we compute the edge density of the CC on either side of this branch. As explained before, the length of each MST branch is the right distance threshold to partition a graph into CC's, and the edge density of each CC is computed based on all their edges shorter than the threshold. Recall that the Kruskal algorithm builds the MST by sequentially examining all edges in the increasing order of length. When a new MST branch is identified, all the edges that contribute to the edge densities of the CC's on either side of this new branch must have already been examined by the algorithm. Thus, we can compute the edge density by simply keeping track of the number of edges and nodes in each CC thus far examined by the Kruskal algorithm, before it is linked by a new MST branch. The time complexity of the second step of Procedure 2.4 is the same as the Kruskal algorithm, which is $O(e \log e)$ where $e$ is the number of edges in $P(\mathcal{V}, \mu)$. The last step of Procedure 2.4, i.e. Cluster Formation, uses the pre-computed edge densities to determine if a CC is a similar cluster. Since there are at most $|\mathcal{V}| - 1$ branches in the MST, the time complexity for this step is $O(|\mathcal{V}|)$.

## III. Experimental results

In section III-A, we describe the color histogram feature used to represent each video frame and the web video dataset for experiments. Experimental results for the proposed fast similarity search and clustering are presented in Sections III-B and III-C respectively.

[4]The procedure described here assumes all the edges in $P(\mathcal{V}, \mu)$ are of different length. A slightly more complicated version that handles equal-length edges can be found in Appendix 5A of [29].

### A. Color histogram feature and video dataset

We represent each frame in a video using a feature vector that consists of four 178-bin HSV color histograms, each representing a quadrant of a frame [1]. Two slightly different distance measurements are used on the feature vectors. The $d_c$ metric, as defined below, is used for signature generation:

$$\begin{aligned} d_c(x,y) &\triangleq \sum_{i=1}^{4} d_c^q(x_i, y_i) \\ \text{where } d_c^q(x_i, y_i) &\triangleq \sum_{j=1}^{178} |x_i[j] - y_i[j]| \end{aligned} \tag{21}$$

where $x_i$ and $y_i$ for $i \in \{1, 2, 3, 4\}$ represent the quadrant feature vectors. As we normalize each histogram bin value to be within 0 and 1, the range of $d_c(x, y)$ is from 0 to 8. A slightly modified distance, denoted by $\widehat{d}_c$, is used for computing the signature distance:

$$\begin{aligned} \widehat{d}_c(x,y) &\triangleq \sum_{i=1}^{4} \widehat{d}_c^q(x_i, y_i) \\ \text{where } \widehat{d}_c^q(x_i, y_i) &\triangleq \begin{cases} \sum_{j=1, j\neq c}^{178} \frac{2 \cdot |x_i[j] - y_i[j]|}{2 - x_i[c] - y_i[c]} \\ \qquad \text{if } x_i[c], y_i[c] > 0.5 \\ d_c^q(x_i, y_i), \text{ otherwise.} \end{cases} \end{aligned} \tag{22}$$

When $\widehat{d}_c$ is used, if there exists a color $c$ that occupies more than 50% of both histograms, its bin value will be removed, and the $d_c$ metric will be computed over the rest of the histograms after re-normalization. The range of $\widehat{d}_c$ is also between 0 and 8. We have found such a distance measurement to be effective in differentiating image contents with identical background color, such as mathematical plots. Even though $\widehat{d}_c$ and $d_c$ are closely related to each other, unlike $d_c$, $\widehat{d}_c$ is not a metric in the mathematical sense. The use of a metric space is one of the key assumptions behind the ViSig method and the fast similarity search schemes described in Section II-B. As such, we design $\widehat{d}_c$ in such a way that its use can be treated as a post-processing step in the similarity search. It can be shown that $\widehat{d}_c(x, y)$ is always larger than or equal to $d_c(x, y)$:

$$\widehat{d}_c(x,y) \geq d_c(x,y) \tag{23}$$

for any color histogram feature vectors $x$ and $y$ [29, Appx. 3A]. In a similarity search, we are interested in finding the set of "similar" feature vectors whose distances with the query are within some $\epsilon > 0$. Equation (23) implies that the set of similar feature vectors identified by $\widehat{d}_c$ must be a proper subset of the set identified by $d_c$. Thus, we can treat the use of $\widehat{d}_c$ as a post-processing step to refine the results of the similarity search obtained via $d_c$. For the rest of the paper, we adopt this model, and develop the theory of similarity search by assuming the use of a true metric.

All the experiments reported in this paper are based on the dataset of signatures generated from the 46,331 web video clips crawled from the web, with total duration of about 1,800 hours. We refer to this test dataset of signatures as SIGDB. Based on our prior experimental results in [1], we use $m = 100$ seed vectors for each signature, and the top $m' = 6$ ranked signature vectors for computing the signature

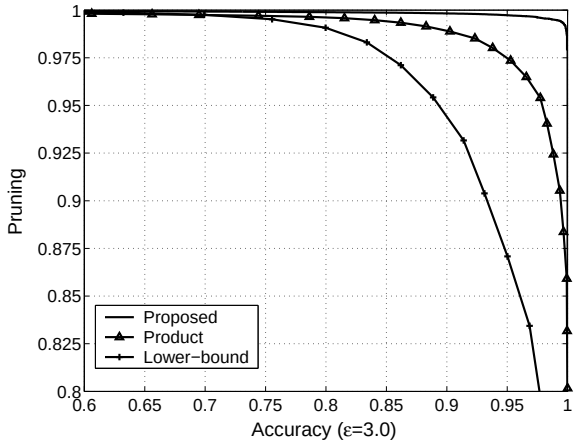


Fig. 2. Pruning-versus-Accuracy plots for the "lower-bound", the "product" and the proposed schemes.

distance as described in Equation (3). The seed vectors are based on keyframes sampled from the video sequences in the database. $\epsilon_C$ used in computing the ranked function (2) is set to be 2.0.

### *B. Experiments on fast search*

Three sets of experiments are performed to characterize the performance of the fast similarity search system described in Section II-B. The first two experiments focus on the pruning-accuracy performance of different components of the feature extraction mapping described in Section II-B.2. The final experiment measures the computation speed for different similarity search schemes.

In the first experiment, we compare the projection vector mapping described in Equation (9) using $l_2$-metric with (a) the "lower-bound" scheme based on the mapping $\mathcal{T}^*(\cdot)$ in Equation (10) and the $l_\infty$-metric, and (b) the "product" scheme based on $\mathcal{T}^*(\cdot)$ and the $\beta(\cdot)$ function defined in (17). It should be noted that the "lower-bound" scheme is the same as the Triangle-Inequality Pruning (TIP) proposed in [22]. We extend all these schemes from handling a single signature vector to the full signature by applying Procedure 2.1. A random query set of 1000 signatures are drawn from the SIGDB and the goal is to identify all signatures in the SIGDB whose signature distances are within $\epsilon = 3.0$ of the queries. Pruning and accuracy values for each scheme, as defined in (8) and (7) respectively, are measured for different $\epsilon'$. The resulting plots of pruning versus accuracy are shown in Figure 2. A good feature extraction mapping should achieve pruning and accuracy that are as close to one as possible. As shown in the figure, our proposed scheme clearly out-performs both the "lower-bound" and "product" schemes by achieving much higher pruning at the same accuracy level. Also, as expected, the "product" scheme out-performs the "lower-bound" scheme as the "product" scheme exploits both the upper and lower bounds of the triangle inequality.

In the second set of experiments, we combine projection vector mapping and PCA as described in Procedure 2.3, and compare its pruning-accuracy performance at different target dimensions with the following schemes:

PCA While in our proposed scheme, PCA is applied on the projection vectors, it can also be directly applied onto the 712-dimensional color histogram feature vectors. In this scheme, we apply PCA to reduce the dimension of the color histograms, and use $l_2$-metric on the resulting range vectors. The use of PCA on the color histogram can be justified as follows: even though PCA is only optimal for $l_2$ metric, the $d_c$ metric used between the color-histogram feature vectors can be bounded above and below by $l_2$ as follows[5]:

$$\frac{1}{\sqrt{712}} \cdot l_2(x_s, y_s) \leq d_c(x_s, y_s) \leq l_2(x_s, y_s) \quad (24)$$

It is thus conceivable to use the $l_2$ metric to approximate the $d_c$ metric.

Fastmap In [18], Faloutsos and Lin have proposed a feature extraction mapping called the Fastmap to approximate a general metric space with a low-dimensional $l_2$ space. Fastmap is a randomized algorithm which iteratively computes each dimension of a range vector by projecting the data onto the "axis" spanned by the two points with maximal separation. $l_2$ distance is used to compare range vectors.

Haar Another method for feature extraction in color histograms is to apply a Haar wavelet transform on the histogram bin values according to the adjacency of colors in the color space. The index vector consists of the few low-pass coefficients from the transform. The Haar wavelet approach used in this paper is based on the scheme described in the MPEG-7 standard [23]. $l_1$ metric is used to compare range vectors.

To compute an appropriate feature extraction mapping for a database, our proposed method and PCA need to scan the entire database once. Fastmap requires multiple scans to find maximally separated data points in the database. The simplest technique is Haar as it is a fixed transform and does not depend on the data at all.

We follow the same procedure as the first experiment to measure accuracy and pruning for all the schemes being tested. Since most of the schemes require training data to generate the mappings, we arbitrary split SIGDB into two halves – we call one half the "training" SIGDB, which is used for building the mapping, and the other half the "testing" SIGDB, which is used for the actual testing. In order to ensure the suitability of incorporating these schemes into GEMINI, we focus on using very low dimensions for range vectors. We test all the schemes for dimensions two, four and eight. The corresponding pruning-accuracy plots are shown in Figures 3(a) through (c). Our proposed scheme results in the best performance in all the dimensions tested, followed by Haar, Fastmap and PCA. The gain of the proposed scheme over the

[5]The proof of the inequality can be found in [16]. Following the same convention as in Equation (12), the $l_2$ metric is normalized by the dimension of the vector.

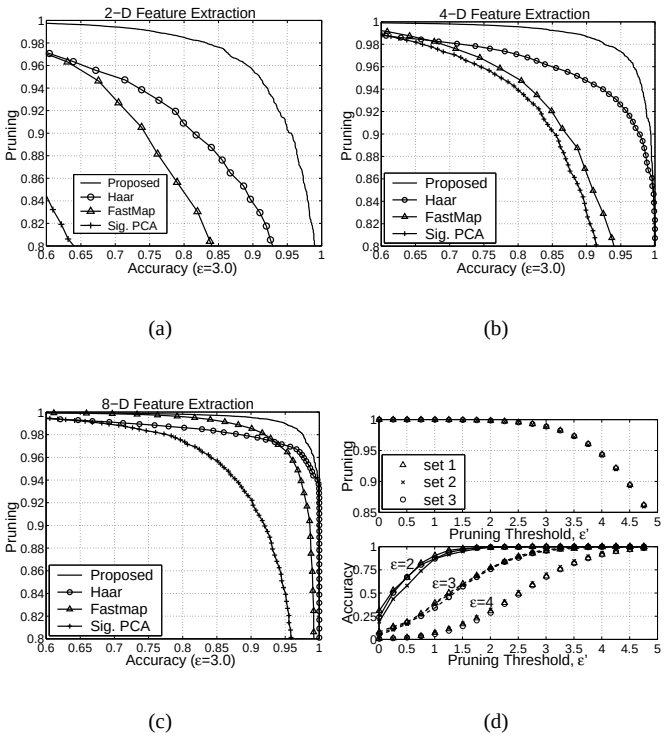


Fig. 3. (a)-(c) Pruning-versus-accuracy plots for two-, four- and eight-dimensional spaces. (d) Pruning and Accuracy versus pruning threshold for three independent sets of queries.

second best scheme, however, diminishes as the dimension increases.

In applying the feature extraction scheme in a fast similarity search, we need to choose a particular value of pruning threshold $\epsilon'$ in order to compute the candidate set. Given the target dimension, accuracy, and pruning, one possible approach is to set $\epsilon'$ to a value that attains the particular level of performance in a previously completed experiment. Thus, an important question to answer is whether the relationship between $\epsilon'$ and the corresponding pruning and accuracy holds for queries other than the ones being tested. To answer this question, we repeat the experiments on our proposed scheme at dimension eight for three independent sets of random queries. Each set has 1000 signatures randomly drawn from the testing SIGDB. For each set of queries, different values of pruning and accuracy are measured by varying $\epsilon'$. The experiment is also repeated for three different values of $\epsilon$, namely 2, 3, and 4. The resulting plots of pruning and accuracy versus $\epsilon'$ for the three query sets and different values of $\epsilon$ are shown in Figure 3(d). As shown in the figure, there is little variation in the amount of pruning among the three sets. There is some variation in the accuracy for small $\epsilon$, but the variation diminishes as $\epsilon$ becomes larger. The maximum differences in accuracy among the three sets over all possible values of $\epsilon'$ are 0.12, 0.06, and 0.04 for $\epsilon = 2$, 3, and 4 respectively. These fluctuations are small compared to the high accuracy required by typical applications.

In the final experiment, we perform a number of speed measurements on a particular platform. Rather than measuring the performance of the entire GEMINI system, we make some simplifications so as to focus on measuring the performance of various feature extraction techniques. The most noted simplification is the absence of a SAM structure in our implementation. The primary function of a SAM structure is to provide fast similarity search on the low-dimensional range vectors. In our implementation, we have chosen to replace it with a simple sequential search. However, we measure time for sequential search separately so that we can compare different schemes fairly. Another function provided by a SAM structure is the memory management for large databases. When a database is too large to fit within the main memory, and hence must be stored in a disk system, a SAM structure typically stores similar data items in contiguous regions within the disk. This can minimize the number of slow random disk access during a similarity search. As the size of our test dataset is moderate, we fit the entire database of signatures and indices within the main memory so as to eliminate the need for memory management.

TABLE I

RESULTS OF SPEED TEST FOR VARIOUS SCHEMES.

| Scheme | Accuracy | Index (ms) | Refine (ms) | Candidate |
|---|---|---|---|---|
| Sequential | 1.00 | - | 6730±35 | - |
| Proposed | 0.89 | 131±0.8 | 33±8 | 109±27 |
| Fastmap | 0.91 | 131±1.5 | 75±11 | 262±39 |
| Haar | 0.92 | 152±1.3 | 123±28 | 428±97 |
| PCA | 0.89 | 130±1.4 | 401±75 | 1386±257 |

We perform our experiments on a Dell PowerEdge 6300 Server with four 550MHz Intel Xeon processors and 1 Gigabytes of memory. As all the tests are run under a single thread, only a single processor is used. The testing SIGDB, which contains 23,206 video signatures, each consisting of 100 signature vectors, and their corresponding 8-dimensional indices are first loaded inside the memory. 100 queries are randomly sampled from the testing SIGDB, and the time to perform the similarity search for each query is measured. Pruning thresholds are chosen, based on the previous experiments, to hit the 90% accuracy level for similarity searches at $\epsilon = 3.0$. As a reference, we also measure the performance of sequential search on signatures with no feature extraction. The results are tabulated in Table I. The "Index" column contains the time required for the sequential search on range vectors to identify the candidate sets. The averages and their standard error at 95% confidence interval are shown. As the Sequential scheme does not use any range vectors, no number is reported. The proposed scheme, Fastmap, and PCA all use the $l_2$ distance on range vectors and thus, result in roughly the same index time. Haar requires slightly larger index time for its $l_1$ distance computation. The "Refine" column is the time required to perform the full signature distance computations on the candidate sets. Our proposed scheme outperforms all other feature extraction schemes in refinement time. The large standard error in the refinement time is due to the variation in the size of candidate sets, as depicted in the last column of the table. Combining the index time and refine time, the proposed

scheme is roughly 41 times faster than the sequential search on signatures.

### C. Experiments on clustering

In this section, we present experimental results on the performance of various clustering algorithms and the statistical analysis of similar clusters found in the web video dataset described in Section III-A. First, we compare the retrieval performance of the proposed algorithm with those of simple-thresholding, single-link, and complete-link techniques. The retrieval performance is measured based on how well an algorithm can identify a set of manually-selected similar video clusters, or the ground-truth set from the dataset. The ground-truth set consists of 443 video sequences in 107 non-overlapping clusters. The size of the clusters ranges from 2 to 20, with average at 4.1. Details of this ground-truth set can be found in [29, ch. 2]. Distances between signatures in all the tested algorithms are computed by the fast similarity search scheme described in Section II-B, with pruning threshold $\epsilon'$ set at 3.0. Since signature distances are stored as graph edges in an edge database, we limit the maximum signature distance stored to 4.0 in order to keep the edge database to a reasonable size. Based on our experience with the ground-truth set construction, this is a reasonable distance threshold as histograms of images that are similar in color are rarely more than 4.0 apart.

The performance metrics used in our experiments are average recall and average precision, which are defined using the concepts of relevant and return sets. Let $\Upsilon$ be the set of all the video sequences in the ground-truth set, and $\mathcal{C}_i, i = 1, \ldots, N$ be the $N$ similar clusters inside. For any query video clip $q \in \Upsilon$, we define the relevant set to $q$ as $\text{rel}(q) = \mathcal{C}_i \setminus \{q\}$ where $q \in \mathcal{C}_i$ and $\setminus$ denotes set exclusion. For any retrieval algorithm, we define the return set of $q$, $\text{ret}(q, \lambda)$, to be the set of video clips returned by an algorithm, excluding $q$ itself, after presenting $q$ as a query. $\lambda$ denotes the parameter(s) used by the algorithm to effect the tradeoff between recall and precision. For simple thresholding, the parameter is a distance threshold and the return set contains all the signatures that are within that particular distance threshold of $q$. The rest of the algorithms are all clustering algorithms and the return set is the cluster that contains $q$[6]. The algorithm parameters are the maximum allowable distance between clusters for single-link and complete-link, and the edge density threshold $\gamma$ for our proposed scheme. Based on the notions of relevant and return sets, we define average recall and average precision of an algorithm as follows:

$$\begin{aligned} \text{Recall}(\lambda) &\triangleq \frac{1}{|\Upsilon|} \sum_{q \in \Upsilon} \frac{|\text{rel}(q) \cap \text{ret}(q, \lambda)|}{|\text{rel}(q)|} \\ \text{Precision}(\lambda) &\triangleq \frac{1}{|\Upsilon|} \sum_{q \in \Upsilon} \frac{|\text{rel}(q) \cap \text{ret}(q, \lambda)|}{|\text{ret}(q, \lambda)|}. \end{aligned} \tag{25}$$

[6]For the general case when $q$ is not part of the database, we can define an ad-hoc distance between $q$ and each cluster based on, for example, the minimum of the distances between $q$ and each of the elements in the cluster. The return set will simply be the cluster closest to $q$.

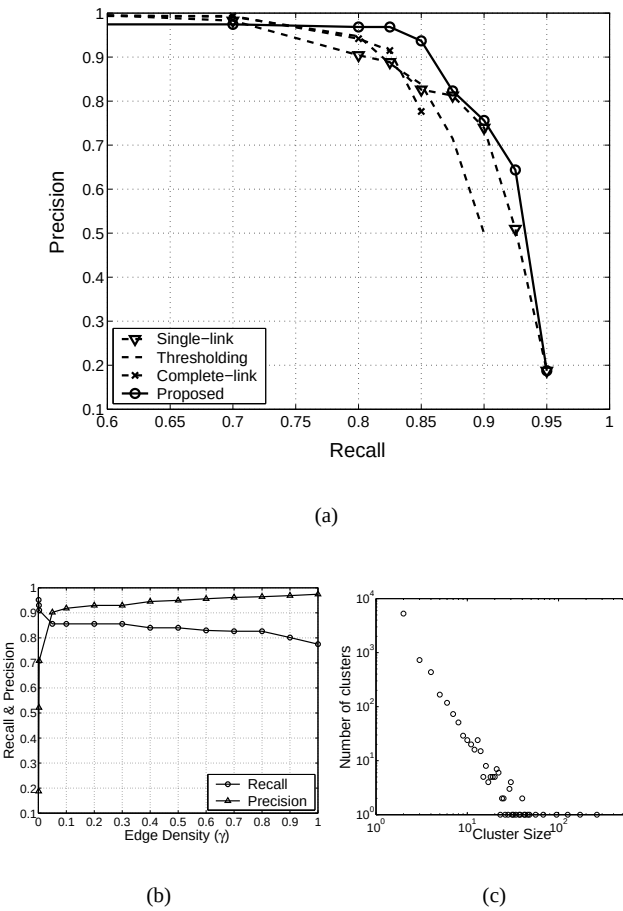


Fig. 4. (a) Precision versus recall for different clustering algorithms and simple thresholding. (b) Precision and Recall versus edge density threshold $\gamma$. (c) Distribution of cluster sizes.

By adjusting the parameters for each of the four algorithms, we measure the average precision values at a pre-defined set of average recall levels. In the case when an algorithm cannot achieve a target recall level precisely, the average precision value reported corresponds to that of the smallest achievable recall level larger than the target level [31]. The resulting precision-versus-recall curves are shown in Figure 4(a). All four algorithms produce similar precision measurements for recall up to 0.7. The precision values for the proposed scheme are slightly lower than the others. This is due to a number of misclassification in some of the smaller clusters in the ground-truth set. The misclassification can be explained as follows: for small CC's, their edge densities as defined in Equation (20) can be large even if they are not exactly "densely" connected. For example, the edge density of a merely connected three-node graph is 0.5. In a much larger CC, an edge density of 0.5 implies that each node is connected to, on average, half of the other nodes in the same CC. This is typically a very good indicator of a similar cluster. As the same edge density threshold is applied to all clusters regardless of their size, it is more likely for the proposed algorithm to misclassify small clusters.

The precision of the proposed algorithm stays roughly the

TABLE II
THE TEN LARGEST CLUSTERS IDENTIFIED IN THE DATASET.

| Label | Size | Diversity | Descriptions or Explanations |
|---|---|---|---|
| A (x) | 263 | 0.12 | Share a segment of red text only |
| B | 172 | 0.70 | Dancing Baby from "Ally McBeal" |
| C (x) | 126 | 0.43 | Share a segment of "MTV News" sign |
| D (x) | 95 | 0.01 | Share a segment of "chv.net" sign |
| E | 68 | 0.01 | An error message from chv.net server |
| F | 56 | 0.98 | Angry man hitting a computer |
| G | 48 | 0.19 | Mathematical plots of wave equation |
| H | 46 | 0.09 | President Clinton televised testimony |
| I | 43 | 0.42 | SOHO Astronomical Video |
| J | 42 | 0.08 | Synthetic Aperture Radar Video |

same for recall up to 0.825 while all the others begin to drop at recall of 0.7. The maximum difference in precision between the proposed algorithm and the rest of the algorithms is about 0.13 at recall of 0.85. We are unable to obtain precision values for simple-thresholding and complete-link at recall values beyond 0.9 and 0.85 respectively because the distance thresholds for both algorithms have reached the maximum distance value stored in the edge database. For recall values beyond 0.85, the precision of the proposed algorithm drops rapidly as chain-like clusters begin to form due to the low edge density threshold. At this point, the proposed algorithm behaves very similar to the single link algorithm. None of the algorithms can achieve perfect recall as some of the similar video sequences in ground-truth clusters have different color content and produce very large color histogram distances. Other types of features must be used in order to identify these similar video sequences.

Figure 4(b) shows the plots of precision and recall versus the edge density threshold $\gamma$ of our proposed algorithm. Larger $\gamma$ values correspond to more strict criterion in forming clusters, leading to larger precision and smaller recall. The retrieval performance stays roughly constant for a large range of $\gamma$. As a result, the detection of similar clusters in a large web video dataset should be relatively insensitive to the choice of $\gamma$.

To further study how similar video clips are distributed on the web, we set $\gamma$ to 0.2, and produce a clustering structure on our experimental database. The resultant clustering structure has a total of 7,056 clusters, with average cluster size of 2.95. Since there are 46,331 video clips in the database, $7056 \cdot 2.95/46331 \approx 45\%$ of the video clips in our database have at least one possibly similar version. Figure 4(c) shows the distribution of cluster sizes. The majority of the clusters are very small – 25% of the clusters have only two video clips in them. Nonetheless, there are a few clusters that are very large. The abundance of similar versions in these clusters may indicate that these video clips are very popular in the web community.

Table II lists the top ten clusters identified in the clustering structure. We provide each cluster with a label in the first column for ease of reference. The "x" sign next to a label indicates that less than 95% of the video clips in that cluster are similar to each other based on manual inspection. The second column indicates the size of each cluster. In the third column, we consider the diversity of web locations where similar video clips are found. We notice that it is quite common for a content creator to put similar video clips such as those compressed in different formats on the same web-page. Diversity is the ratio between the number of distinct web-pages in each cluster and the cluster size. A large ratio implies that video clips in that cluster are originated from very diverse locations.

As shown in Table II, clusters A, C and D contain some wrongly classified video clips. Upon careful examination, we have found that all the video clips in each of these clusters share a visually-similar segment, from which multiple signature vectors are selected. As a result, they are classified to be "similar" even though the remainder of the content in these clips are very different. A possible remedy to this problem is to raise the required number of similar signature vectors shared between two signatures before declaring them as similar. Among those which are correctly classified, clusters E, G, H and J consist of clips that are mostly from the same web-page. Some of them are identical, such as the error message found in cluster E. Others have very similar visual contents such as those in G, H and J. These types of sequences are generated intentionally by the content creators, and provide little information on how video sequences are copied and modified by different web users. On the other hand, clusters B, F, and I have relatively high diversity values. Video sequences in cluster B are from a popular television show, cluster F contains a humorous sequence of a man's frustration towards his computer, and cluster I contains astronomical video clips from a large-scale, multi-nation research project. Large clusters with high diversity values seem to indicate that the corresponding video content is of interest to a large number of web users. Such information can be used to provide better ranking for retrieval results – popular content should be ranked higher as they are more likely to be requested by users.

## IV. CONCLUSIONS AND FUTURE WORK

We have described a video search engine that uses video signatures to represent video clips from a large database such as the web. The two major algorithmic designs in this search engine are a feature extraction mapping for fast similarity search, and a clustering algorithm for grouping signatures into similar clusters. In our proposed feature extraction mapping, we have made use of the squared signature distances between the signature vectors and the seed vectors to form a projection vector. The dimension of the projection vectors is further reduced by using PCA. We have shown experimentally that this technique provides better trade-off between accuracy and pruning as compared with PCA, Fastmap, and Haar Wavelet. Even though the use of projection vectors has been motivated based on experimental data of color histograms, the concept is based on the triangle inequality which is a defining property of any metric space. As such, we plan to test this technique on other types of feature vectors for video similarity detection. Another direction is to study the effect of seed vectors on the performance of similarity search. The proposed feature extraction mapping is based on distances between signature vectors

and the seed vectors, which are randomly sampled from a training dataset [1]. It is conceivable that more sophisticated methods can be used to select better seed vectors so that the resulting mapping produces better trade-off between accuracy and pruning. A study on the similar approach has shown that the similarity search performance can indeed be improved by carefully choosing the seed vectors[7] to match the data [22].

We have also proposed a graph-theoretical algorithm that identifies highly-connected CC's formed at different distance thresholds as clusters. The algorithm is based on the classical Kruskal algorithm on building minimum spanning tree. The criterion to determine a cluster has been based on an experimentally-determined edge density threshold $\gamma$. As the goal of the clustering algorithm is to combat the uncertainty in distance measurement, a better characterization of the imprecision in ViSig method and dimension reduction should provide a more direct method for choosing $\gamma$. Another issue that we have not addressed in this paper is how to maintain the clustering structure when video signatures are inserted or deleted from the database. Based on the current design, we can first update the sorted edge database, then re-build the minimum spanning tree, and finally re-cluster based on the new tree. The update of the edge database can be carried out efficiently using the B-tree index in the Berkeley DB package. On the other hand, re-building the MST runs in linear time with respect to the number of edges and re-clustering in linear time with respect to the number of nodes. One possible approach to speed up the last two steps is to perform a local repair on the MST in the case when the new signatures affect only a small portion of the graph.

## ACKNOWLEDGMENT

The authors thank the anonymous reviewers for providing valuable comments on the initial drafts of this paper. For S.-C. Cheung, part of the revision of this paper was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

## REFERENCES

[1] S.-C. Cheung and A. Zakhor, "Efficient video similarity measurement with video signature," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 1, pp. 59–74, Jan. 2003.

[2] C. Silverstein, M. Henzinger, J. Marais, and M. Moricz, "Analysis of a very large altavista query log," Compaq Systems Research Center, Tech. Rep. SRC-Technical Note 1998-014, 1998.

[3] M. Naphade, R. Wang, and T. Huang, "Multimodal pattern matching for audio-visual query and retrieval," in *Proceedings of the Storage and Retrieval for Media Datbases 2001*, vol. 4315, San Jose, USA, jan 2001, pp. 188–195.

[4] D. Adjeroh, I. King, and M. Lee, "A distance measure for video sequence similarity matching," in *Proceedings International Workshop on Multi-Media Database Management Systems*, Dayton, OH, USA, Aug. 1998, pp. 72–9.

[5] R. Lienhart, W. Effelsberg, and R. Jain, "VisualGREP: A systematic method to compare and retrieve video sequences," in *Proceedings of storage and retrieval for image and video databases VI*, vol. 3312. SPIE, Jan. 1998, pp. 271–82.

[6] H. Chang, S. Sull, and S. Lee, "Efficient video indexing scheme for content-based retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, pp. 1269–79, Dec 1999.

[7] P. Indyk, G. Iyengar, and N. Shivakumar, "Finding pirated video sequences on the internet," Stanford Infolab, Tech. Rep., Feb. 1999.

[8] G. Iyengar and A. Lippman, "Distributional clustering for efficient content-based retrieval of images and video," in *Proceedings 1998 International Conference on Image Processing*, vol. III, Vancouver, B.C., Canada, 2000, pp. 81–4.

[9] H. Greenspan, J. Goldberger, and A. Mayer, "A probabilistic framework for spatio-temporal video representation," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.

[10] N. Vasconcelos, "On the complexity of probabilistic image retrieval," in *Proceedings Eighth IEEE International Conference on Computer Vision*, vol. 2, Vancouver, B.C., Canada, 2001, pp. 400–407.

[11] H. Samet, *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989.

[12] C. Faloutsos, *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, 1996.

[13] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*. Morgan Kaufmann, 1997, pp. 426–435.

[14] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the 24th International Conference on Very-Large Databases (VLDB'98)*, New York, NY, USA, Aug. 1998, pp. 194–205.

[15] J. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. of Educational Psychology*, vol. 24, pp. 417–441, 1933.

[16] G. Golub and C. van Loan, *Matrix Computation*, 3rd ed. The Johns Hopkins University Press, 1996.

[17] T. F. Cox and M. A. Cox, *Multidimensional scaling*, 2nd ed. Boca Raton : Chapman & Hall, 2001.

[18] C. Faloutsos and K.-I. Lin, "Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proceedings of ACM-SIGMOD*, May 1995, pp. 163–174.

[19] J. Bourgain, "On lipschitz embedding of finite metric spaces in hilbert space," *Israel Journal of Mathematics*, vol. 52, pp. 46–52, 1985.

[20] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applictions," *Combinatorica*, vol. 15, no. 2, pp. 215–45, 1995.

[21] G. Hristescu and M. Farach-Colton, "Cluster-preserving embedding of proteins," Rutgers University, Piscataway, USA, Tech. Rep. DIMACS 99-50, 1999.

[22] A. P. Berman and L. G. Shapiro, "A flexible image database system for content-based retrieval," *Computer Vision and Image Understanding*, vol. 75, no. 1/2, pp. 175–195, July/August 1999.

[23] L. Cieplinski, S. Jeannin, M. Kim, and J.-R. Ohm, "Visual working draft 4.0," ISO/IEC JTC1/SC29/WG11, Tech. Rep. W3522, July 2000.

[24] A. Broder, S. Glassman, M. Manasse, and G. Zweig, "Syntactic clustering of the web," in *Sixth International World Wide Web Conference*, ser. Computer Networks and ISDN Systems, vol. 29, no.8-13, Sept. 1997, pp. 1157–66.

[25] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 1999.

[26] J. Bezdek, J. Keller, R. Krisnapuram, and N. R. Pal, *Fuzzy models and algorithms for pattern recognition and image processing*. Kluwer Academic Publishers, 1999.

[27] J. Kruskal, "On the shortest spanning subtree of a graph and the travelling salesman problem." *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, 1956.

[28] C. Zahn, "Graph-theoretcial methods for detecting and describing gestalt clusters," *IEEE Transactions on Computers*, vol. 20, no. 1, pp. 65–86, January 1971.

[29] S.-C. Cheung, "Efficient video similarity measurement and search," Ph.D. dissertation, University of California, Berkeley, 2002.

[30] T. Cormen, C. Leiserson, and R. Riverst, *Introduction to Algorithms*, 2nd ed. Cambridge, Massachusetts: The MIT Press, 2001.

[31] , "Common evaluation measures," in *The Eleventh Text Retetrieval Conference (TREC 2002)*, ser. NIST Special Publication 500-251, 2002, http://trec.nist.gov/pubs/trec11/appendices/MEASURES.pdf.gz.

[7] The term "key" was used in the original paper.

**Sen-ching S. Cheung** Sen-ching S. Cheung received the B.S. degree (summa cum laude) from University of Washington, Seattle in 1992. From 1995 to 1998, he was a research engineer at Compression Labs Inc, San Jose, California where he did research and development work in video processing and network protocols for professional video conferencing systems. He was also actively involved in ITU-T Study Group and MPEG standard activities. He completed his Ph. D. at U.C. Berkeley in 2002. Currently, he is a post-doctoral researcher at the Center of Applied Scientifi c Computing in Lawrence Livermore National Laboratory. His research interests are in the general area of image and video processing, content-based multimedia retrieval, pattern recognition, and scientifi c data mining.

**Avideh Zakhor** Avideh Zakhor received the B. S. degree from California Institute of Technology, Pasadena, and the S. M. and Ph. D. degrees from Massachusetts Institute of Technology, Cambridge, all in electrical engineering, in 1983, 1985, and 1987 respectively. In 1988, she joined the Faculty at U. C. Berkeley where she is currently Professor in the Department of Electrical Engineering and Computer Sciences. Her research interests are in the general area of image and video processing, compression, and communication. Together with her students, She has won a number of best paper awards, including the IEEE Signal Processing Society in 1997, IEEE Circuits and Systems Society in 1997 and 1999, international conference on image processing in 1999, and Packet Video Workshop in 2002. She holds 5 U.S. patents, and is the co-author of the book, "Oversampled A/D Converters" with Soren Hein.

Prof. Zakhor was a General Motors scholar from 1982 to 1983, was a Hertz fellow from 1984 to 1988, received the Presidential Young Investigators (PYI) award, and Offi ce of Naval Research (ONR) young investigator award in 1992. From 1998 to 2001, she was an elected member of IEEE Signal Processing Borad of Governers. In 2001, she was elected as IEEE fellow.